Getting Started With Shared Library In Jenkins

Apr 30, 2024

Why Shared Library

Shared library help us to archive the DRY (Do Not Repeat) concept. Means that there might be scenarios where you need to repeat some steps you can use Shared Library to write your code there that can be used in more then one job.

How To Setup For Shared Library

To Setup a shared library in Jenkins you need 2 configuration one is the repository structure and second is the adding the repository in the Jenkins configuration and you need to have some knowledge of groovy.

Setting The Repo

First we have to setup our source code management repository and we have to create a directory name `vars` any groovy script written in this directory will act as module and any written function will act as submodule. To use the same in a Jenkinsfile we have to import the library and we will be using them like `module.submodule`.

For demo purpose here is a sample for the same in repository.

```
.
.......
└── vars
    ├── lib_module_1.groovy
    └── lib_module_2.groovy
```

Here I have created a directory name `vars` as it's required then based on your requirement we can have as much as modules we want I have added 2 modules in it `lib_module_1` and `lib_module_2`.

In `lib_module_1` I have written a function named as `func_from_module_1`.

```
def func_from_module_1(){
    println("This is a test function from module 1 and function func_from_module_1")
    println("This function can be used in any pipeline stages")
}
```

In `lib_module_2` I have written a function named as `func_from_module_2`.

```
def func_from_module_2(){
    println("This is a test function from module 1 and function func_from_module_2")
    println("This function can be used in any pipeline stages")
}
```

These functions are just printing some text on the console.

Now these names `lib_module_2` or `func_from_module_2` can be anything you want.

Adding The Configuration In The Jenkins

To add the above repository as a library we have to add the same in Jenkins and to do the same here are the options.

`Manage Jenkins -> Configure System -> Global Pipeline Libraries -> Add`

As you hit add button you will see a few options here are a few of them.

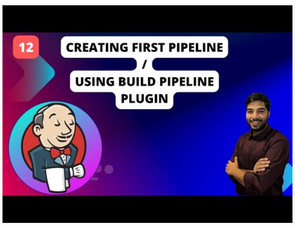
`Name` : The name of the new Shared Library.

`Default version` : Set a default brach of the repository to be used as library.

`Allow default version to be overridden` : This option allows scripts to select a custom version of the library.

How To Use The Same In Jenkinsfile/Jenkins Pipeline

Before we start the same make sure you know how to write Jenkinsfile for more on the same you can checkout this video.



To use the same in Jenkinsfile we needs to use the `@Library` annotation before stating the `pipeline` block and we have to pass the library name to the same. Or we can use the same as a Jenkinsfile step like `library 'test_library'`.

Using the configured version of the library.

```
@Library('test_library') _
```

or if using in step

```
library 'test_library'
```

Using any other version of the library let's say `branch_01`.

```
@Library('test_library@branch_01') _
```

or if using in step

```
library 'test_library@master'
```

Using multiple library in a single pipeline.

```
@Library(['test_library', 'test_library@other_version']) _
```

or if using in step

```
library 'test_library, test_library@other_version'
```

Example 1:

```
@Library('test_library@branch_01') _
pipeline{
    agent 'any'
    stages{
        stage("Stage_Name"){
            steps{
                sh "echo Test Stage.; pwd; hostname"
                script{
                    lib_module_1.func_from_module_1()
                }
            }
        }
    }
    post{
        success{
            cleanWs()
        }
    }
}
```

Example 2:

```
pipeline{
    agent 'any'
    stages{
        stage("Stage_Name"){
            steps{
                library 'test_library'
                sh "echo Test Stage.; pwd; hostname"
                script{
                    lib_module_1.func_from_module_1()
                }
            }
        }
    }
    post{
        success{
            cleanWs()
        }
    }
}
```